VZDU # 28 JAN/FEB 1991

DOWN UNDER CLUB

Editor
Harry Huggins
12 Thomas Str.
Mitcham. 3132.
03-873 1408

Treasurer
Ron Allen
2 Orlando str.
Hampton.3188.
03-598 4534

I am sorry if this issue is a little skimpy, and will probably be a little late as well. Don't jump to conclusions about a Xmas hangover.

Fact is I had a fall off a ladder, and that has slowed me up a "little". I do only use 2 fingers to type, but I like one on each hand, and my right arm is in a sling and not of much use.

We had quite a gathering last meeting. We were able to welcome Paul Frantz from North Q'ld. No doubt down here to look at dry ground for a change.

Remember our meetings are the first Sunday of the month at 12 Thomas Str. MITCHAM. But if you are in Melb. anytime drop in.

Also missing is the 3rd article on VZ SOUND. Bob is either having a holiday, or else gone down with the floods. What a state!!!

Dave Mitchell's home is in Rockhampton. We hope he did'nt fare badly. We have'nt heard. ood luck Dave.

I draw attention to the series by David Wood on Mystry program writing. This is an excellent series, and well worth following, if not for Mystry writing then for the use of Basic.

The name changing routine for disk users is covered in BME in VZDU #15 starting at line 7570. That is one method. There are others. When the series is finished I will cover faster saving of Data to tape.

Harry

DEATH MAZE

```
10 REM THIS PROGRAM WAS WRITTEN
20 REM IN  1988  BY  BEN HOBSON
30 REM AND  TAKES   UP  4.932 K
40 REM
45 JX$="JOYSTICK":KB$="KEYBOARD"
50 V=3:JY=0:W=1:S=0:HS=0:GOSUB1180:GOTO1280:'CHUCK GOSUB 1180 ::
60 MODE(1):GOTO390
70 IFW=1THENX1=0:Y1=1:X=5:Y=5
80 IFW=2THENX1=0:Y1=1:X=56:Y=29
90 IFW=3THENX1=0:Y1=1:X=58:Y=25
100 COLOR2:FORA=116TO126:SET(A,62):NEXT:COLOR3
105 COLOR2:FORB=1TO20:RESET(X,Y):FORG=1TO20:NEXT:SET(X,Y)
106 FORG=1TO20:NEXT:NEXT
110 SOUND16,1;18,1;20,1;23,3;20,1;23,3
120 A=RND(63):B=RND(127):COLOR4:SET(B,A):COLOR2:S=S+10
125 FOR FG=0 TO DY: NEXT FG
130 H$=INKEY$:IFJY=1THENGOTO940
140 IFH$="A"THENY1=1:X1=0:GOTO180
150 IFH$="Q"THENY1=-1:X1=0:GOTO180
160 IFH$="M"THENX1=-1:Y1=0:GOTO180
170 IFH$=","THENX1=1:Y1=0:COLOR2
180 SET(X,Y):X=X+X1:Y=Y+Y1:IFPOINT(X,Y)=1THENSET(X,Y):GOTO120
190 IFY=62ANDX=116THEN320
200 IFY=62ANDX=117THEN320
210 IFY=62ANDX=118THEN320
220 IFY=62ANDX=119THEN320
230 IFY=62ANDX=120THEN320
240 IFY=62ANDX=121THEN320
250 IFY=62ANDX=122THEN320
260 IFY=62ANDX=123THEN320
270 IFY=62ANDX=124THEN320
280 IFY=62ANDX=125THEN320
290 IFY=62ANDX=126THEN320
300 IFPOINT(X,Y)=2THENGOTO850
310 GOTO850:GOTO120
320 SOUND31,1:CLS:FORJ=1TO10:PRINT"    *** CONGRATULATIONS ***"
330 CLS:PRINT"    *** CONGRATULATIONS ***":CLS:NEXT
340 PRINT:PRINT:PRINT:PRINT
350 PRINT"YOU MADE IT THROUGH SCREEN ";W
360 IFW=3THENGOTO1890
370 PRINT"NOW TRY SCREEN ";W+1:W=W+1
380 FORK=1TO2000:NEXT
390 IFW=1THENGOTO630
400 IFW=2THENGOTO420
410 IFW=3THENGOTO1430
420 MODE(1):COLOR3:W=2
430 FORA=0TO127:SET(A,0):SET(A,63):NEXT
440 FORB=0TO63:SET(0,B):SET(127,B):NEXT
450 FORC=0TO47:SET(30,C):SET(100,C):NEXT
460  FORD=31TO63:SET(90,D):SET(115,D):NEXT
470 FORE=0TO42:SET(80,E):NEXT
480 FORF=47TO80:SET(F,43):NEXT
490 FORG=11TO19:SET(G,8):NEXT
500 FORH=8TO51:SET(19,H):NEXT
510 FORI=19TO47:SET(I,51):NEXT
520 FORJ=13TO51:SET(47,J):NEXT
```

```
530 FORK=47TO68:SET(K,13):NEXT
540 FORL=13TO35:SET(68,L):NEXT
550 FORM=68TO53STEP-1:SET(M,35):NEXT
560 FORN=35TO23STEP-1:SET(53,N):NEXT
570 FORO=53TO59:SET(O,23):NEXT
580 FORP=23TO32:SET(59,P):NEXT
590 FORQ=8TO19:SET(10,Q):NEXT
600 FORR=10TO13:SET(R,19):NEXT
610 FORT=19TO51:SET(13,T):NEXT
620 GOTO70
630 MODE(1):W=1:COLOR3:FORA=0TO127:SET(A,0):SET(A,63):NEXT:B=0
640 A=0:B=0:C=0:D=0:E=0:F=0:G=0:H=0:I=0
650 FORA=0TO63:SET(0,A):SET(127,A):NEXT
660 FORA=14TO1STEP-1:B=B+1:SET(A,B):NEXT
670 FORA=30TO1STEP-1:C=C+1:SET(A,C):NEXT
680 FORA=45TO1STEP-1:D=D+1:SET(A,D):NEXT
690 FORA=60TO1STEP-1:E=E+1:SET(A,E):NEXT
700 FORA=75TO14STEP-1:F=F+1:SET(A,F):NEXT
710 FORA=90TO29STEP-1:G=G+1:SET(A,G):NEXT
720 FORA=110TO49STEP-1:H=H+1:SET(A,H):NEXT
730 FORA=125TO63STEP-1:I=I+1:SET(A,I):NEXT
740 FOR A=40TO63:SET(115,A):NEXT
750 FOR A=40TO50:SET(87,A):NEXT
760 A=11:B=4:FORJ=1TO5: RESET(A,B):A=A-1:B=B+1:NEXT
770 A=17:B=14:FORJ=1TO5: RESET(A,B):A=A-1:B=B+1:NEXT
780 A=40:B=6:FORJ=1TO5:RESET(A,B):A=A-1:B=B+1:NEXT
790 A=70:B=6:FORJ=1TO5:RESET(A,B):A=A-1:B=B+1:NEXT
800 A=105:B=6:FORJ=1TO5:RESET(A,B):A=A-1:B=B+1:NEXT
810 A=20:B=41:FORJ=1TO5:RESET(A,B):A=A-1:B=B+1:NEXT
820 A=50:B=41:FORJ=1TO5:RESET(A,B):A=A-1:B=B+1:NEXT
830 A=83:B=43:FORJ=1TO5:RESET(A,B):A=A-1:B=B+1:NEXT
840 GOTO70
850 IFV>1THENSOUND4,2;8,2;9,2;11,9:S=S-1000
860 POKE30744,0:CLS:PRINT:CLS:V=V-1
870 FORJ=1TO6:PRINT:NEXT
880 IFV=2THENPRINT"*2 LIVES LEFT - PREPARE TO DIE*"
890 IFV=1THENPRINT"*THIS IS YOUR LAST RIDE SUCKER*"
900 IFV=0THENPRINT"    ** TOO BAD YOU DIED **
910 IFV=0THENGOSUB1240
920 IFV=0THENFORG=1TO2000:GOTO1280
930 FORK=1TO2000:NEXT:GOTO60
940 K=(INP(46AND43)AND31)
950 IFK=29THENY1=1:X1=0:GOTO180
960 IFK=30THENY1=-1:X1=0:GOTO180
970 IFK=27THENX1=-1:Y1=0:GOTO180
980 IFK=23THENX1=1:Y1=0:GOTO180
990 COLOR2:SET(X,Y)
1000 X=X+X1:Y=Y+Y1
1010 IFPOINT(X,Y)=3THEN850
1020 IFPOINT(X,Y)=1THEN120
1030 IFPOINT(X,Y)=4THEN850
1040 CLS:POKE30744,1
1050 CLS:PRINT"   I N S T R U C T I O N S
1060 PRINT:PRINT"IN DEATH RACE YOU HAVE TO ":PRINT"NEGOTIATE";
1070 PRINT" YOUR WAY AROUND THE   MAZE WITHOUT HITTING";
1080 PRINT" YOUR OWN  LINE , THE BOUNDARIES , OR ANY  DOTS";
1090 PRINT" OR YOU WILL DIE. THERE ARE THREE SCREENS.
1100 PRINT:PRINT"FUNCTION   KEYS    JOYSTICK"
```

```
1110 PRINT"UP           <Q>      UP
1120 PRINT"DOWN         <A>      DOWN
1130 PRINT"LEFT         <M>      LEFT
1140 PRINT"RIGHT        <,>      RIGHT
1150 PRINT:PRINT"       PRESS <C> TO CONTINUE";
1160 X$=INKEY$:X$=INKEY$:IFX$<>"C"THEN1160
1170 GOTO1280
1180 CLS:FORG=1TO5:PRINT:NEXT
1190 PRINT"        D E A T H - R A C E      ";
1200 FORG=1TO4:PRINT:NEXT
1205 PRINT"              VERSION II"
1210 PRINT"         WRITTEN IN 1990 BY       ";
1220 PRINT"              BEN HOBSON          ";
1230 GOTO1240:RETURN
1240 IFS>HSTHENHS=S
1250 FORA=1TO3:SOUND4,4:FORD=1TO12:NEXTD,A:SOUND7,3:FORD=1TO12
1260 NEXT:SOUND6,1:FORD=1TO12:NEXT:SOUND6,2:FORA=1TO2:SOUND4,2
1270 FORD=1TO12:NEXTD,A:SOUND3,3;4,6:RETURN
1280 V=3:W=1
1285 CLS:PRINT" * * * D E A T H  R A C E * * *"
1290 PRINT@71,"< > SELECT"
1295 PRINT@112,"SELECTED"
1300 PRINT@162,"<0> TO <9>  DIFFICULTY LEVEL"
1305 PRINT@236,"(0) HARD"
1310 PRINT@268,"(9) EASY"
1315 PRINT@328,"<I> INSTRUCTIONS
1320 PRINT@392,"LAST SCORE 00000"
1325 PRINT@424,"HIGH SCORE 00000"
1330 PRINT@491,"PRESS  KEY";
1335 IFJY=1PRINT@103,JX$:PRINT@72,"K":PRINT@82,KB$:SOUND10,1
1340 IFJY=0PRINT@103,KB$:PRINT@72,"J":PRINT@82,JX$:SOUND20,1
1345 A$=INKEY$:A$=INKEY$:IFA$=""THEN1345
1350 IFA$="J"THENJY=1:GOTO1335
1355 IFA$="K"THENJY=0:GOTO1335
1360 IFA$="I"THENPOKE30744,0:GOTO1050
1361 Q=ASC(A$)-48
1365 IFQ>=0ANDQ<=9THENDY=(Q*20):S=0:GOTO60
1370 GOTO1345
1375 :
1380 :
1385 :
1390 :
1430 MODE(1):W=3:COLOR3
1440 FORQ=0TO127:SET(Q,0):SET(Q,63):NEXT
1450 FORQ=0TO63:SET(0,Q):SET(127,Q):NEXT
1460 FORQ=1TO20:SET(Q,18):NEXT
1470 FORQ=1TO18:SET(20,Q):NEXT
1480 FORQ=1TO25:SET(Q,25):NEXT
1490 FORQ=1TO19:SET(Q,36):NEXT
1500 FORQ=1TO58:SET(Q,58):NEXT
1510 FORQ=43TO62:SET(115,Q):NEXT
1520 FORQ=12TO89:SET(Q,52):NEXT
1530 FORQ=68TO104:SET(Q,58) :NEXT
1540 FORQ=37TO58:SET(104,Q):NEXT
1550 FORQ=104TO122:SET(Q,37):NEXT
1560 FORQ=22TO37:SET(122,Q):NEXT
1570 FORQ=96TO122:SET(Q,22):NEXT
1580 FORQ=18TO22:SET(96,Q):NEXT
```

```
1590 FORQ=96TO115:SET(Q,18):NEXT
1600 FORQ=0TO18:SET(115,Q):NEXT
1610 FORQ=12TO83:SET(Q,43):NEXT
1620 FORQ=4TO43:SET(83,Q):NEXT
1630 FORQ=83TO102:SET(Q,4):NEXT
1640 FORQ=4TO15:SET(102,Q):NEXT
1650 FORQ=87TO102:SET(Q,15):NEXT
1660 FORQ=15TO25:SET(87,Q):NEXT
1670 FORQ=87TO107:SET(Q,25):NEXT
1680 FORQ=25TO30:SET(107,Q):NEXT
1690 FORQ=89TO107:SET(Q,30):NEXT
1700 FORQ=30TO52:SET(89,Q):NEXT
1710 FORQ=12TO43:SET(33,Q):NEXT
1720 FORQ=12TO33:SET(Q,32):NEXT
1730 FORQ=28TO32:SET(Q,12):NEXT
1740 FORQ=12TO21:SET(28,Q):NEXT
1750 FORQ=28TO33:SET(Q,21):NEXT
1760 FORQ=0TO16:SET(60,Q):NEXT
1770 FORQ=0TO8:SET(43,Q):NEXT
1780 FORQ=43TO60:SET(Q,8):NEXT
1790 FORQ=43TO74:SET(Q,17):NEXT
1800 FORQ=17TO36:SET(43,Q):NEXT
1810 FORQ=17TO36:SET(74,Q):NEXT
1820 FORQ=51TO66:SET(Q,32):NEXT
1830 FORQ=23TO32:SET(66,Q):NEXT
1840 FORQ=23TO32:SET(51,Q):NEXT
1850 FORQ=22TO28:SET(55,Q):NEXT
1860 FORQ=55TO61:SET(Q,22):NEXT
1870 FORQ=22TO28:SET(62,Q):NEXT
1880 GOTO70
1890 PRINT" AT LEVEL";DY/20
1900 PRINT
1910 PRINT"NOW TRY ALL 3 AGAIN AT LEVEL ";
1920 IFDY/20=0PRINT"0":GOTO1940
1930 DY=DY-20:PRINTDY/20
1940 FORZ=1TO5000:NEXT:W=1:GOTO390
```

THE SCREAM SHEET

By The Official SCREAMER

EVERYTHING'S GOING SCREAMINGLY.

Hee, hee, heeee. I've done it! I've finally stumped that little machine in the end room. There it sat, its screen blank with confusion; just as it has caused me to sit so often, MY mind a blank. Before I tell you what I did, let me first relate to you the "Saga of my Second Computer".

Late in 1989 I decided to up-grade my qualifications. I was working as a driver and yearned for a more interesting occupation, but I had a problem——no qualifications, no special experience. So in 1990 I began a part-time external (correspondence) university course. With the aid of the VZ-300 and Word-pro cartridge I worked over-time. I pruned 4,500 word essays down to the required 2,000 words. I printed and reprinted. I saved, saved again, merged and resaved...all on tape. And thereby hangs this tale.

Saving, verifying and loading from tape was agonisingly slow. Midnight came and went with monotonous regularity. As well as my course I still had two sons, a husband and a house to care for and I still had a five hour a day driving job. My eyes were wearing out faster than my tyre treads and printer ribbons. What to do? Here's where "Dear Harry..." came back to the rescue.

After receiving my letter about needing a second computer, because my sons were complaining that a certain mother was monopolising THEIR computer ("After all Mum, we have assignments to do too!" "And high scores to beat," his brother added) "Dear Harry..." replied that a second-hand VZ-300 was available. Not only that, but disc drive, power pack and etceteras were also available. I jumped at it——my very own computer, complete with disc drive——but I had an end-of-year assignment to put in, so it sat in the corner for a month.

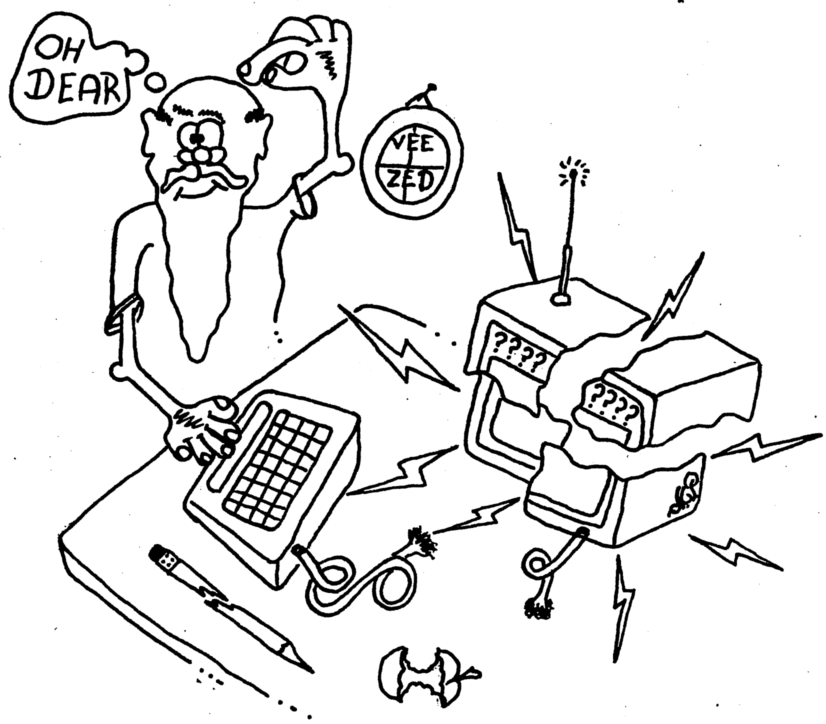
The day came to unpack, hook-up and revel in the glory of total ownership, BUT...what's that!! NO COLOUR and it won't work on cassette! How will I transfer my records to disc? What'll I do? Take too long for a letter...blow the phone bill...ring, rring..."Hey Harry, it's me!" Yes, Harry to the rescue again, this time with the help of another VZ-er, Nifty Nev. Nev pulled it apart, said "Oh dear" and told Harry. Harry said "Oh dear" and promptly despatched a replacement computer to my post box.

A WHOLE FORTNIGHT LATER. (Another phone bill blowout.) "Er...Harry, it's me. No, it hasn't arrived. Didn't you send it?" A trip to town to the P.O. "No, we haven't received it. No, there's no record of it in the book. It can't have arrived." "No, madam, it's no good looking in the cupboard. Yes, it's the cupboard with the door open." "This box, madam? VZ-300 computer box with your address on it, madam? Here you are...but madam, it's NOT IN THE BOOK."

Speed home. Unpack it, hook it up, load a tape. Great...but...oh no...where's the colour going? Nev fiddled with the colour thingummy, said it must have gotten jiggled in-transit and home I went again. And that's when it happened.

I connected up the cassette to load a tape to transfer to disc, then I connected up the disc drive and turned the computer on. The screen went blank. It lost its little tail-wagging cursor. A row of question marks appeared along the top of the screen. IT DIDN'T KNOW WHAT TO DO! Hee, hee, hee...sob, sob...HA-A-ARRY !!!

WOT? No brickbats???



Dear Editor,

I was rather interested in the Letter to the Editor in #26, Sept/Oct, 1990 edition of the newsletter.

It looked like a bit of fun and something unusual, so I was very disappointed to progress no further than "OUT OF DATA ERROR IN 10" which was the very first line.

Please repeat this exercise so that I might be able to join in the fun and read the letter.

regards,
"Mystified."

SEE PAGE 12
VZDU #27
ADD LINES 200 ON
(Ed)

Athol Wedgewood writes:-

Accolades for the VZDU Database - it abounds with uses. I used it to list, group and value household items following a serious fire. Reserving different first characters for each item category, followed by alph. sorting simplified the task. The insurance assesor was very impressed with the printout, especially the running total of the amounts, and this enabled a fast settlement of the claim involving nearly 1000 items.

Athol.

P.S. Thanks for a great newsletter.

GAMES COLUMN

Paul has not written this column, as he has been down here in Victoria. No doubt to have a look at a state that is not half under water!!!

This issue we have an influx of new High Scores. One I don't understand, but am printing it. Maybe Matthew can enlighten me or Paul.

We have the second part of How to write Mystry Programs, from David Wood. Also, standing by for a later issue another program from Peter Ross.

There is a good Maize program in this issue. It is not the "Run of the mill" up-down, left-right, but in a diagonal format, that makes it really hard.

The HIGH Scores

DAWN PATROL	52500	David Wood
CRASH	410	Harry Huggins
DIG OUT	24400	Paul Frantz
HAMBURGER SAM	39500	Stephen Frantz
LADDER CHALLENGE	22530	Peter Watso
KAMIKAZE	11160	Peter Watson
TEN PIN BOWLS	185	David Wood
VZ INVADERS	28080	Peter Watson
GALAXON	150000	Nicole Huggins
PENGUIN	1350	Jason Oakley
LUNAR LANDER	3600	Jason Oakley
SUPER SNAKE	811	Tim Oliver
MAZE OF ARGON	73258	Peter Watson
ASTEROIDS	11000	Peter McLean
CIRCUS	1210	Peter Watson
PANIK	7700	Peter Watson
HOPPY	25550	Matthew McLean
GHOST HUNTER	23400	Chris McLean
STAR BLASTER	480 units left	Matthew McLean

Matt. I don't understand the 480 units left.

Peter. I think we leave the typing out. A typist would come up with 80-90 WPM or more.

INSTALLING A LOAD/SAVE FEATURE FOR ADVENTURE PROGRAMS

In many adventures the players find that as they are nearing completion of the set task, they need to go away and do something else, like eat, or sleep. When they do so they either leave the computer running and return to find the power supply making an alarmingly loud buzzing noise and warm enough to burn fingers when touched (with the computer itself not much cooler), or they switch off and their progress disappears forever. The most obvious solution to this problem is to enable them to save the data relevant to their progress to tape or disk where it can be loaded back into the program at a later date.

The command best suited to this on the VZ is the PRINT# command, which is unfortunate, because it doesn't work! (except on the old VZ200 datasette, and even in this case extremely slowly.)

Fortunately for disk drive users the DOS equivalents of the PRINT# and INPUT# commands do work, so you won't have to mess around trying to find the right memory loaction to move your data to, but as I am not a disk drive owner, I am unsure of how quick or efficient these are. To save the data the following is used:

```
7000 CLS: PRINT"INSERT DISK IN DRIVE":GOSUB 970
7010 OPEN "ADVDATA",1
7020 PR# "ADVDATA",R,S,W
7030 FOR I=1 TO 35: PR# "ADVDATA",F(I): NEXT
7040 FOR I=1 TO 17: PR# "ADVDATA",C(I): NEXT
7050 CLOSE "ADVDATA"
7060 RETURN
```

Note that the subroutine at 970 place the message "PRESS ANY KEY TO CONTINUE....." at the bottom of the screen and waits until the user has pressed a key before RETURNing. If you have two disk drives you will have to use a message like "INSERT DATA DISK IN DRIVE 2" and select the appropriate drive using the DRIVE command.

To load the data again the following lines would be used:

```
7100 CLS: PRINT"INSERT DATA DISK IN DRIVE": GOSUB 970
7110 OPEN "ADVDATA",0
7120 IN# "ADVDATA",R,S,W
7130 FOR I=1 TO 35: IN# "ADVDATA",F(I): NEXT
7140 FOR I=1 TO 17: IN# "ADVDATA",C(I): NEXT
7150 CLOSE "ADVDATA"
7160 RETURN
```

One problem with saving the data to disk with any method is that any particular filename can only appear once on any disk. So once you have saved a file to disk using the filename ADVDATA you cannot save any other file with the same name. The only possible solutions I can think of to this is to ERASE any old files of the same name before saving, or alternatively, more experienced programmers could ask the player to give his or her file a name, find exactly where the filename appears in the program code in RAM, and POKE the new filename into these locations. (Note that the command SAVE A$ will NOT work.)

```
10 PRINT "THIS IS A SAMPLE MESSAGE......"
20 CLEAR 30
30 PRINT "INPUT A NEW MESSAGE ":INPUT"(30 CHARS. MAX.)";A$
32 IF A$ = "" THEN STOP
```

```
35 'MAKE LENGTH OF USER MESSAGE EQUAL TO 30 CHARS
40 IFLEN(A$)<30 THEN A$=A$+".": GOTO 40
50 FOR I=1 TO 30
55 'GET ASCII VALUE OF NEXT CHARACTER IN MESSAGE
60 A = ASC( MID$(A$,I,1))
65 'AND POKE IT INTO MEMORY LOCATION
67 '(31471 IS THE LOCATION OF THE FIRST 'T' IN 'THIS')
70 POKE 31470+I,A
75 'GO BACK FOR NEXT CHARACTER IN THE MESSAGE
80 NEXT
85 'CLEAR THE SCREEN AND GO BACK TO PRINT THE NEW MESSAGE
90 CLS: GOTO 10
```

This program demonstrates how it is possible for a program to alter itself. If you intend to type it in, make sure you type line 10 exactly as it appears. You can leave out the REM (') statements if you wish. When you run the program, if you wish to stop typing in messages, just press RETURN at the prompt. LIST the program and note how line 10 has altered. The same method could be applied to changing the filename of the adventure data, but you may have some difficulty finding where the SAVE and LOAD statements appear in RAM. (Cassette owners may also consider using this method so that the files on the tape can be easily distinguished from one another.)

The solution to the PRINT# bug for cassette owners is to move the data to a position in high memory, and then save this section of memory to tape. To reload the data, the reverse process is used - the data is loaded into high memory from tape and then returned to the necessary arrays and variables.

Before you attempt to install this feature you should have your program otherwise completely finished and saved to tape or disk. IF YOU MAKE AN ERROR YOU COULD DESTROY THE CONTENTS OF YOUR PROGRAM. The program must be finished as you need to be able to keep track of the end of BASIC at all times.

The position in memory where you store this data depends on your own memory size and what you want to do with your program. If you want the program to be for your own use only, the end of your data should be just over a quarter of a kilobyte (256 bytes) below your computer's top of memory.

MEMORY LOCATIONS OF TOP OF MEMORY
Unexpanded VZ300 - 47103 (-18433) expanded VZ200 - 53247 (-12289)
expanded VZ300 - 63487 (-2049) VZ200 +28K/VZ300+18K - 65535 (-1)
(Note that each memory location above 32767 must have 65536 subtracted from it before it can be addressed by BASIC's PEEK and POKE commands. For example if you wanted to POKE a value of 42 into location 53247 you would type either POKE 53247-65536,42 or POKE -12289,42. however if you wanted to PEEK at the value in memory location 28672, which is below 32767, you would type PRINT PEEK(28672).)

If, however, you require the program, for use on several different configurations of VZ - for example, if you wished to send your program in for use on one of VZ Down Under's public domain tapes - you should choose the lowest top of memory value location possible, provided that there is at least about one and a half kilobytes spare. (HINT: To determine the amount of free memory, run your program until the first location screen appears, then press BREAK and type POKE 30862,212:POKE 30863,39: PRINT USR(0). This calls the MEM routine, which is one the

you will get 31465 which is the Start Of Basic. If you tried this again with the End Of Basic pointer (30969 and 30970) you will get 31469 if you don't have a program in memory, and a higher value if you do. (In some cases the value will be above 32767 so you might need to subtract 65536.)

Therefore what you need to do to save your data block to tape is place the memory location where your block starts into the Start Of Basic pointer, The memory location after where your block finishes (IE the final location PLUS one) into the End Of Basic pointer, CSAVE the block with the desired filename and then restore these pointers to their original values.

To find the values to put in the pointers, the location of the start of your data block should be divided by 256. The whole number part of the result is your Most Significant Byte. Then take the value of the MSB, multiply it by 256 and subtract this value from the original location number. The result of this is your Least Significant Byte. For example if the start of your data block is 52931, divide this by 256 and your result will be 206.76563. Therefore in this case the MSB is 206. For the LSB, (206*256) is subtracted from 52931 and the result is 195. Therefore the value POKEd into 30884 is 195 and 30885 is 206. A similar process is used to determine the values to be POKEd into the End Of Basic Pointer - 30969 and 30970. The location of the end of your data block PLUS ONE should be used. For example, if the end of your data block is 52986, you should use a value of 52987 for your EOB pointer - the MSB (POKEd into 30970) is 206 and the LSB (POKEd into 30969) is 251.

After this, the next step is to save the data. To do this simply use the CSAVE "FILENAME" command. Then you must restore the values in the pointers to what they were before you changed them. Unless you have done something really tricky with your program, your SOB (Start Of Basic) should be 31465 - a value of 233 in 30884 and a value of 122 in 30885. However with the EOB things get a bit more complicated. Every time you change your program and make it either shorter or longer, and the value in the EOB changes. This is why it is important not to install this feature until your program is otherwise finished, as you will need to change the value you restore to the EOB pointer every time you alter the program. The best way to overcome this is to add your SAVE and LOAD routines using dummy values for 30969 and 30970. (EG 7070 POKE 30969, 000: POKE 30970,000). THEN after you have finished, PEEK at 30969 and 30970 then replace the dummy values with the real ones (NOTE: if you get a value such as 76, you will have to replace it as POKE 30969,076, because you will otherwise shorten the program.)

This is the sample save routine in full.

```
7025 CLS: PRINT"   PRESS RECORD ON THE TAPE": GOSUB 970
7027 REM 970 - PRESS ANY KEY...
7030 POKE 30884,195: POKE 30885,206
7040 POKE 30969,251: POKE 30970,206
7050 CSAVE "ADVDATA"
7060 POKE 30884,233: POKE 30885,122
7070 POKE 30969,108: POKE 30970,182
7075 REM SOUND A BEEP. (DO THIS LATER)
7080 R$="STOP THE TAPE":' R$ IS THE MESSAGE RETURNED TO THE PLAYER.
7090 RETURN
```

If you wish to load your data block into memory, all you have to do is CLOAD it. However if you want to do this from inside the program

BASIC commands left dormant on the VZ. this gives the number of bytes of memory free.) For example, if you own an expanded VZ300 and you find that the amount of free memory was 18393, you would place your data at a position ending a quarter of a kilobyte below 47103. However, if the value was fairly close to 16384, you would need to place your data about a quarter of a kilobyte below 53247. The actual routine to move the data into high memory is fairly simple.

```
7000 F(35)=R: F(36)=S: F(37)=W
7010 FOR I=1 TO 38:POKE 52931 - 65536 +I,F(I): NEXT
7020 FOR I=1 TO 17:POKE 52969 - 65536 +I,C(I): NEXT
```

The important variables R,S and W (which are the player's current location, strength, and amount of weight carried) are placed into the F array. The contents of the F array are POKEd into the memory locations 52932-52969 (minus 65536 of course!) by the means of a simple loop. Likewise the contents of the C array are POKEd into locations 52970-52986. (Don't worry too much about what the F and the C arrays do for the moment - they will be studied in more detail later on. Basically, the C array records the locations of the objects the player can pick up, and the F array records the actions a player has carried out - if a particular door has been opened, or a hidden object made visible.) Note that in this particular instant the data is loaded to the top of memory of an expanded VZ200 - 261 bytes below the actual top of memory (53247). Also note that because the data must be POKEd into various memory locations, the maximum value of any element of the F and C arrays cannot exceed 255, but programmers should not find this a major limitation.

When a successful load is made from tape, the data is returned to the memory locations from which it is saved. Therefore the routine which places this data into the appropriate variables and arrays is simply the "reverse" of the routine which moves the data to high memory.

```
7150 POKE 31070,0: FOR I=1 to 38: F(I) = PEEK (52931 - 65536 +I): NEXT
7160 FOR I=1 to 17: C(I)= PEEK (52969 - 65536 +I): NEXT
7170 R=F(35): S=F(36): W=F(37)
7180 RETURN
```

The data is PEEKed into the F and C arrays using two loops. The values of R,S and W are regained from the F array.

The next trick is to fool your VZ into thinking your data is a BASIC program so it will save it to tape as one. (If you have a copy of the article "Memory Dumps To Tape" which appeared in VZ Down Under # 7 and also LE'VZ newsletter # 16 it would probably be a good idea to refer to it.) When the VZ saves a program it simply saves everything in between the memory location specified in the Start Of Basic pointer and that specified by the End Of Basic pointer, sometimes also known as the start of Simple Variables Table pointer. For those of you who have not come across a pointer before, it consists of two bytes (the values in memory locations 30884 and 30885, which is the Start Of Basic pointer, for example.) The lower value contains the LEAST SIGNIFICANT BYTE (LSB) of the address being pointed to and the higher value contains the MOST SIGNIFICANT BYTE (MSB). When you multiply the value of the MSB by 256 and then add the value of the LSB you will get the value being pointed to. For example, if you PEEKed at the values in 30884 and 30885, on most occassions the values returned will be 233 and 122 respectively. When 122 is multiplied by 256 and 233 is added

there are a couple of complications. Firstly the CLOAD command clears all of the variables, and secondly it ends the program run. A marginal improvement is the CRUN command which will still clear the variables but will restart the program afterwards. Also both commands will leave the EOB and two other pointers - the Start of Dimensioned Variables Table (DVT - 30971/2) and the End of Variables Table (30973/4) - with the address of the end of the data block. Because the Top Of Memory (TOM - 30897/8) has been lowered below this value in order to protect your data block, the result will be an ?OUT OF MEMORY ERROR. The solution to this is to raise the TOM and another pointer, the Top Of Stack (TOS) back to their original values, CLEAR 0 to stabilise the other pointers, CRUN your data block, then restore the SOB, EOB, DVT and EVT pointers to their correct values (at the start of the program, because it has restarted after the CRUN command), before once again lowering TOM and TOS. The values to be POKEd into the TOM and TOS can be obtained by using your own particular value of TOM and the same calculations as before. IE

MSB = INT(address/256)
LSB = address - (MSB * 256)

The TOS should be kept 50 bytes below the TOM so if your MSB for the TOM is 207 and the LSB is 255, for the TOS, the MSB and LSB will be 207 and 205 respectively.

As the program restarts, there will be no variables in use, so the DVT and EVT will hold exactly the same values as the EOB. As a rough guide for lowering the TOM and TOS, if your data block is less than about 250 bytes long (most will be) simply lower the MSB of both the TOM and TOS by 2.

Lastly for this routine as the program will restart after loading, the program will need to know whether it is just being RUN or whether it has restarted after a CRUN command. This can be done by POKEing a certain value into a certain memory location, and then just after the program restarts checking this location. I chose to POKE 42 into location 31070 (an unused address in the Communications Region) but you may choose different values and locations if you wish. This is the sample LOAD routine

```
7100 POKE 31070,42:CLS: PRINT "START THE TAPE.": GOSUB 970
7110 POKE 30897,255: POKE 30898, 207
7120 POKE 30880,205: POKE 30881, 207
7125 CLEAR 0000
7130 CRUN "ADVDATA"
7135 'PROGRAM RESTARTS AFTER THIS

START OF PROGRAM

2 POKE 30884,233: POKE 30885,122
3 POKE 30969,108: POKE 30970,182
4 POKE 30971,108: POKE 30972,182
5 POKE 30973,108: POKE 30974,182
6 POKE 30897,255: POKE 30898,205
7 POKE 30880,205: POKE 30881,205
11 CLEAR 700
.
.
.
INITIALISATION
.
.
```

```
25 IF PEEK (31070)=42 THEN GOSUB 7150
27 ' LOAD DATA FROM MEMORY INTO VARIABLES
```

Note that you will need to put dummy values into lines 2,3 and 4 until you know what the exact values of the EOB. For those of you who type in the example program, it is unlikely that you will type it exactly as it appears, so you will need to peek 30969 and 30970 when you have finished typing it in and place these values in 2,3,4 and 7070.

With line 25 above, after the initialisation has put the default (starting) values in the C and F arrays, this causes a call to line 7150 which will replace these with the values that were loaded. The program flow then continues so that the player can continue from where the game was saved

Now the moment of truth for those that installed the feature. CSAVE your program again (Don't tape over your last copy) because if you have made a mistake either the system will crash or you will get an ?OUT OF MEMORY ERROR, both of which would have fatal consequences for your program. RUN your program, and move around, carrying out a few different actions so the game status is different from the start. Type SAVE at the prompt, press record on the tape, then any key on the keyboard. Stop the tape after the beep if you decided to one in the program, or when the next location screen appears if you didn't. BREAK the program and rewind the tape to before the start of your file. RUN the program again, typing LOAD at the first prompt, pressing play then any key. The program should restart (Here is the spot where the ?OUT OF MEMORY ERRORs occur) and hopefully a new location screen will appear. If so you have successfully installed a LOAD/SAVE feature to your program!

(Because of the way the feature is installed, there are a few short cuts the adventurer can take with this routine. If you don't want to load the data from within the program, you can CLOAD your data, type POKE 31070,42, load your adventure program, if you haven't already, then run it. Also if you have loaded your progress, and then while you are playing your character bites the dust, you can resurrect it from where you last loaded the progress by BREAKing the program, and simply typing POKE 31070,42 before running it again. The data should still be in memory from the last load and this causes the computer to call the routine at line 7150.)

Well this just about wraps it up for this edition. Later on we will be examinig how this method can be developed so the program can be split into modules so you can create longer adventures which don't all occupy memory at the same time. (For those of you that have the listing of section two of the demonstration program, you can still type it in without knowing how it works. Just place dummy values in lines 6,7,8 and 7180 and replace these with the value of the EOB. To run it you must SAVE your progress at the end of section one and load this as soon as section two tells you to.)

Finally, if you had problems understanding this section, don't worry. Although a LOAD/SAVE feature is useful, it is not essential and you don't have to include it in your adventures. This part is the most complex, and the rest of the series should be much simpler.